Caratteri, stringhe e altri dati

I singoli caratteri si rappresentano usando numeri. Nelle versioni più semplici ogni carattere è un numero senza segno a sette oppure otto bit. Nella rappresentazione più usata per caratteri in varie lingue ogni carattere si memorizza invece come una sequenza di numeri a otto bit, la cui lunghezza varia da uno (otto bit) a sei (quarantotto bit totali).

ASCII

Ogni carattere si rappresenta usando un numero binario di sette bit, oppure otto mettendo il primo bit a zero. Si tratta quindi di numeri che vanno da `0` a `127`. Per esempio:

```
'z' → 1111010 (numero 122)
';' → 0111011 (numero 59)
'E' → 1000101 (numero 69)
'0' → 0110000 (numero 48)
```

Alcuni numeri nell'intervallo `0–127` non rappresentano dei veri e propri caratteri (ossia dei segni tipografici come 'a', ';', '@', ecc.). Per esempio, nelle comunicazioni fra computer e terminali seriali, il numero `19` trasmesso dal terminale indicava che questo era troppo impegnato per ricevere altri dati, e quindi richiedeva la sospensione della comunicazione. Quando riceveva questo carattere speciale, il computer doveva bloccare la comunicazione, trattenendosi dal trasmettere ulteriori dati al terminale. Il numero `17` segnalava la richiesta di riprendere la comunicazione.

La seguente tabella mostra i numeri corrispondenti ai vari caratteri. Come si può vedere, alcuni numeri sono riservati ai caratteri speciali (come appunto `XOFF=DC3=19` e `XON=DC1=17`), mentre mancano del tutto le lettere accentate e altri caratteri di uso comune come il simbolo di euro.

0:	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
10:	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
20:	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
30:	RS	US	SP	!	"	#	$	%	&	'
40:	(	)	*	+	,	-	.	/	0	1
50:	2	3	4	5	6	7	8	9	:	;
60:	<	=	>	?	@	A	B	C	D	E
70:	F	G	H	I	J	K	L	M	N	O
80:	P	Q	R	S	T	U	V	W	X	Y
90:	Z	[	\	]	^	_	`	a	b	c
100:	d	e	f	g	h	i	j	k	l	m
110:	n	o	p	q	r	s	t	u	v	w
120:	x	y	z	{	|	}	~	DEL		

ISO-8859-1

Questo meccanismo estende ASCII mediante l'uso dell'ottavo bit, il più significativo. Ogni carattere viene quindi rappresentato con un numero senza segno a otto bit, ossia un numero nell'intervallo `0–255`. I primi centoventotto numeri coincidono con quelli di ASCII, inclusi i caratteri speciali.

I numeri successivi vengono usati per lettere accentate o straniere come `ò`, `ñ`, `å` e vari altri caratteri come il segno di centesimo `¢`, le virgolette caporali aperte « e chiuse ».

0:	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
10:	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3

20:	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
30:	RS	US	SP	!	"	#	$	%	&	'
40:	(	)	*	+	,	-	.	/	0	1
50:	2	3	4	5	6	7	8	9	:	;
60:	<	=	>	?	@	A	B	C	D	E
70:	F	G	H	I	J	K	L	M	N	O
80:	P	Q	R	S	T	U	V	W	X	Y
90:	Z	[	\	]	^	_	`	a	b	c
100:	d	e	f	g	h	i	j	k	l	m
110:	n	o	p	q	r	s	t	u	v	w
120:	x	y	z	{	|	}	~	DEL		
130:										
140:										
150:										
160:		¡	¢	£	¤	¥	¦	§	¨	©
170:	ª	«	¬		®	¯	°	±	²	³
180:	´	µ	¶	·	¸	¹	º	»	¼	½
190:	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç
200:	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ
210:	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û
220:	Ü	Ý	Þ	ß	à	á	â	ã	ä	å
230:	æ	ç	è	é	ê	ë	ì	í	î	ï
240:	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù
250:	ú	û	ü	ý	þ	ÿ				

Questo sistema va bene per lingue come l'italiano, l'inglese, lo spagnolo, il norvegese e altre, ma copre solo in parte i simboli usati in altre come il turco (in cui esiste una lettera simile alla i ma senza il punto: `ı`). Per queste altre lingue, esistono meccanismi analoghi: ISO-8859-2, ISO-8859-2, ecc. Per esempio, ISO-8859-9 usa il numero `253` per la i senza punto `ı` invece che per la y con accento acuto `ý` come in ISO-8859-1.

Questo sistema presenta però degli svantaggi: ogni volta che si vuole memorizzare una sequenza di caratteri in una certa lingua occorre specificare il codice che si usa; non si possono scrivere testi in più lingue (come quando per esempio si vuole citare un frammento in greco all'interno di un testo in italiano); per ogni nuova lingua da aggiungere va specificato un codice da usare; alcune lingue come il cinese hanno più di 255 caratteri.

Unicode, UTF-8

Universal set code è un meccanismo di codifica dei caratteri che impiega più di otto bit per ogni carattere. Nella versione moderna, ogni carattere viene rappresentato usando trentadue bit. A loro volta, questi trentadue bit possono venire memorizzati in tre modi diversi:

utf-32 (chiamata anche ucs-4)
: un singolo numero a trentadue bit

utf-16
: sequenza fino a 2 unità da 16 bit (è una evoluzione del precedente ucs-2, codifica di lunghezza fissa a 2 byte)

utf-8
: i primi 128 caratteri, che sono gli stessi di ASCII, sono rappresentati come un singolo numero a otto bit il cui primo bit è `0` (quindi ogni carattere ASCII è rappresentato come se stesso); per gli altri caratteri, si usa una sequenza di numeri a otto bit che hanno tutti `1` come primo bit; il primo di questi numeri è

nella forma `1...10...`, e il numero di uni indica la lunghezza della sequenza (da due a sei - in pratica usata solo la sequenza massima di 4 byte).

Questo meccanismo è quello preferenzialmente usato nel software moderno. Per esempio, in un qualsiasi browser web è possibile visualizzare i caratteri "ŧ", "å", "ı", "□".

Stringhe

Le stringhe sono sequenze di caratteri. Ad esempio, in Python è possibile stampare la sequenza `'ciao a tutti!'` con il seguente comando:

```
print 'ciao a tutti!'
```

La sequenza fra virgolette è una stringa, ossia una sequenza di caratteri, ognuno rappresentato in uno dei modi detti sopra.

In genere, i numeri che rappresentano una stringa vengono memorizzati l'uno di seguito all'altro, usando poi il numero zero per indicare la fine della stringa. Questo è possibile perchè in tutti i meccanismi visti sopra, da ASCII a utf-8, il numero zero è riservato a questo scopo: è il carattere speciale `NUL` che indica la fine di una stringa. Nel caso dell'esempio di sopra, in memoria `'ciao a tutti!'` è la sequenza:

```
99 105 97 111 32 97 32 116 117 116 116 105 33 0
```

Come si può vedere infatti dalle tabelle di sopra, `99=c`, `105=i`, `97=a`, ecc. Il numero `32` è lo spazio, il `33` il punto esclamativo e lo zero indica la fine della stringa.

Notare che il numero zero indica la fine della stringa, mentre il carattere `0` viene rappresentato dal numero `48`, come si può vedere dalle tabelle di sopra (le altre cifre sono rappresentate da `49`, `50`, ecc). Per esempio, la stringa `"da 0 a 5"` si rappresenta come `100 97 32 48 32 97 32 53 0`, dato che `100=d`, `97=a`, `32` è lo spazio, `48` la cifra zero, ecc.

Il ritorno a capo viene indicato o con il singolo carattere `10` o con la coppia `10 13`. Storicamente, il numero `10` indicava a una stampante di far avanzare la carta dello spazio di una linea, mentre `13` diceva di riportare il carrello all'inizio della linea.

Questo sistema di indicare la fine della stringa con uno zero è usato in gran parte dei linguaggi di programmazione. Esistono però delle alternative. Una è quella di riservare dello spazio all'inizio per il numero di caratteri che la stringa contiene. La stringa dell'esempio si rappresenta quindi con la sequenza:

```
00013 99 105 97 111 32 116 117 116 116 105 33
```

Il primo numero è a sedici bit invece che a otto come gli altri in modo da poter così venire usato per stringhe lunghe più di 256 caratteri (fino a 65536). Il fatto che il carattere rappresentato con zero non abbia un significato di terminazione permette la sua presenza all'interno della stringa.

Questo sistema si usa in alcuni linguaggi di programmazione, ma il suo principio base di premettere la lunghezza ai dati veri e propri è usato in altri contesti, come la memorizzazione dei file e il trasferimento via web (http). In questi casi, la lunghezza in termini di numero di caratteri viene indicata prima o separatamente dai dati stessi, che possono quindi contenere anche lo zero.

Oltre al sistema della terminazione con zero e della lunghezza all'inizio, esiste un terzo meccanismo di rappresentazione delle stringhe, in cui queste vengono memorizzate con strutture "ad albero" che facilitano l'inserimento o la cancellazione dei caratteri anche in mezzo alla sequenza.

Suoni

I suoni sono onde di pressione dell'aria. Tale pressione può venire misurata a intervalli di tempo regolari (es. 48000 volte al secondo) e il valore ottenuto rappresentato come un numero binario (es. a 16 bit). Suoni rappresentati come una simile sequenza di numeri possono poi venire memorizzati, trasmessi e riprodotti.

La sequenza non rappresenta in modo esatto l'originale per due motivi:

- la pressione viene misurata solo un certo numero di volte al secondo, e quindi variazioni avvenute da una misurazione all'altra non vengono rilevate;
- la pressione è un valore continuo, mentre per poter usare un numero prefissato di bit occorre approssimarlo (vedere la pagina sui numeri in virgola mobile).

Tanto maggiore è la *frequenza di campionamento* (numero di misurazioni al secondo) e il numero di bit usati, tanto più la rappresentazione è fedele all'originale.

La sequenza di numeri può venire poi *compressa* per ridurre lo spazio di memoria che occupa. Esistono poi alcuni formati che rappresentano non suoni in generale ma quelli che è possibile generare da strumenti elettronici.

Immagini

Per memorizzare immagini si sfrutta il fatto la maggior parte dei colori visibili all'occhio umano si possono considerare un miscuglio in quantità variabili di rosso, verde e blu. Per esempio, questo rettangolo è un miscuglio di una intensità massima di rosso, mezza di verde e nulla di blu.

Una immagine, come per esempio una foto, viene quindi memorizzata considerandola composta da un grande numero di piccoli quadrati chiamati *pixel*. Per ogni pixel, si rappresentano le quantità di rosso, verde e blu che contiene.

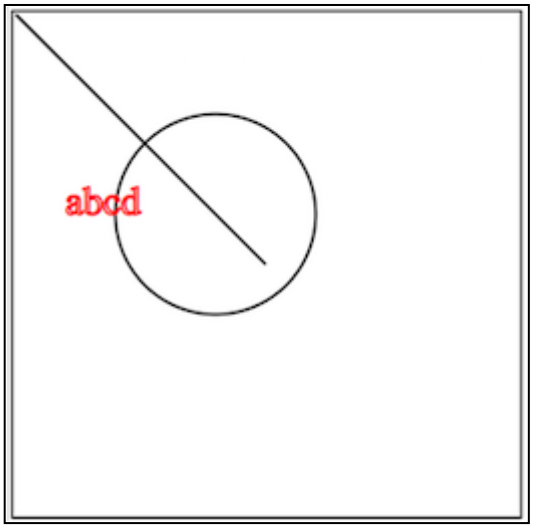
L'immagine complessiva si può poi rappresentare in vari modi. Uno dei formati più semplici, il *ppm*, è fatto in questo modo:

- una parte iniziale che contiene fra l'altro la larghezza e l'altezza dell'immagine;
- la sequenza di valori dei pixel a partire da quello in alto a sinistra; ogni pixel è rappresentato attraverso la sua quantità di rosso, verde e blu.

Come per i suoni, si tratta comunque di una rappresentazione approssimata, per i soliti due motivi: ogni pixel viene considerato di un colore unico (quindi variazioni di colore più piccole di un pixel vengono ignorate), e i colori vengono rappresentati con un numero finito di bit (per cui esiste una perdita di precisione).

Anche per le immagini esistono meccanismi di compressione, che permettono di ridurre lo spazio di memoria richiesto.

Immagini rappresentate come quadrati di pixel si dicono *raster*. Esiste un altro meccanismo, che consiste nello specificare le figure geometriche elementari (linee, cerchi, ecc.) di cui un'immagine è composta. Per esempio:

Immagine	Rappresentazione
	<?xml version="1.0" standalone="no"?> <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"> <svg xmlns="http://www.w3.org/2000/svg"> <g style="stroke:#000000;fill:none;"> <polyline points="0,0 100,100" /> <circle cx="80" cy="80" r="40" /> <text x="20" y="80" stroke="#FF0000">abcd</text> </g> </svg>

La linea `<polyline points="0,0 100,100" />` specifica una segmento dal punto di coordinata `0,0` al punto di coordinata `100,100`. La successiva è un cerchio, quella dopo la scritta in rosso `abcd`. L'origine delle coordinate è nell'angolo in *alto* a sinistra, invece che in basso a sinistra dov'è di solito.

Come si vede da questo esempio, l'immagine è rappresentata attraverso una sequenza di caratteri, da `<?xml` fino a `</svg>`. Invece di specificare il colore in ogni punto, viene detto che l'immagine è composta da un segmento, un cerchio e una scritta in rosso. Immagini specificate descrivendo elementi geometrici sono dette *vettoriali*. Nella maggior parte dei casi, all'interno delle immagini vettoriali si possono inserire come elementi (oltre a linee, cerchi, ecc.) anche delle immagini raster.